REMARKS

Claims 1-29 are pending in the present application. Claims 2, 3, 12, 13, 22, and 23 are amended to replace the trademark word "Java" with the phrase "platform independent" as suggested in the Office Action. Reconsideration of the claims is respectfully requested.

Amendments to the specification are made to capitalize the trademark "JAVA" and to correct a reference number. Also, the specification is amended to include a reference to "DVD 282" as suggested in the Office Action.

I. 35 U.S.C. § 112, Second Paragraph

The Office Action rejects claims 2, 3, 12, 13, 22, and 23 under 35 U.S.C. § 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter, which Applicants regard as the invention. This rejection is respectfully traversed.

As to claims 2, 3, 12, 13, 22, and 23, the Office Action states:

> References to trademarks in claims or using trademarks as part of claims should not be used. The trademarked term refers to the source of the goods and not the technical merits of those goods.

Office Action dated October 27, 2003, page 2.

Claims 2, 3, 12, 13, 22, and 23 are amended to replace the trademark word "Java" with the phrase "platform independent" as suggested by the Office Action. Therefore the rejection of claims 2, 3, 12, 13, 22, and 23 under 35 U.S.C. § 112, second paragraph has been overcome.

II. 35 U.S.C. § 102, Alleged Anticipation Based on McManis

The Office Action rejects claims 1, 2, 4-6, 11, 12, 14-16, 21, 22, 24-26 under 35 U.S.C. § 102(b) as being allegedly anticipated by McManis (U.S. Patent Number 5,692,047). This rejection is respectfully traversed.

As to independent claims 1, 11 and 21, the Office Action states:

> Regarding Claims 1, McManis teaches: A method of verifying the integrity of unauthenticated code, comprising:
>
> a. receiving automatically authenticated code, the automatically authenticated code including an embedded first hash value of the unauthenticated code (col. 10 lines 64-col. 11 lines 1-2);
>
> b. receiving the unauthenticated code; generating a second hash value of the unauthenticated code (col. 11 lines 2-9);
>
> c. comparing the first hash value and the second hash (col. 11 lines 10-25);
>
> d. verifying the integrity of the unauthenticated code if the first hash value and the second hash value match (col. 11 lines 10-25).
>
> ...
>
> Claims 11, 12, 14-16, 21, 22, 24-26 are rejected for the same reasons outlined above.

Office Action dated October 27, 2003, pages 4-5.

Independent claim 1, which is representative of the other rejected independent claims 11 and 21 with regard to similarly recited subject matter, reads as follows:

> 1. A method of verifying the integrity of unauthenticated code, comprising:
>
> receiving automatically authenticated code, the automatically authenticated code including an embedded first hash value of the unauthenticated code;
>
> receiving the unauthenticated code;
>
> generating a second hash value of the unauthenticated code;
>
> comparing the first hash value and the second hash value; and
>
> verifying the integrity of the unauthenticated code if the first hash value and the second hash value match.

A prior art reference anticipates the claimed invention under 35 U.S.C. § 102 only if every element of a claimed invention is identically shown in that single reference, arranged as they are in the claims. *In re Bond*, 910 F.2d 831, 832, 15 U.S.P.Q.2d 1566, 1567 (Fed. Cir. 1990). All limitations of the claimed invention must be considered when determining patentability. *In re Lowry*, 32 F.3d 1579, 1582, 32 U.S.P.Q.2d 1031, 1034 (Fed. Cir. 1994). Anticipation focuses on whether a claim reads on the product or process a prior art reference discloses, not on what the reference broadly teaches. *Kalman v. Kimberly-Clark Corp.*, 713 F.2d 760, 218 U.S.P.Q. 781 (Fed. Cir. 1983). Applicants respectfully submit that *McManis* does not identically show every element of the claimed invention arranged as they are in the claims. Specifically, *McManis* does not teach the feature of receiving automatically authenticated code, the automatically authenticated

code including an embedded first hash value of the unauthenticated code. Additionally, *McManis* does not teach that the first hash value is embedded in the automatically authenticated code.

McManis is directed towards a system and method in which a program interpreter for programs, whose integrity is verifiable, includes a facility for using non-verifiable programs from trusted sources. In addition, *McManis* provides a system and method for refusing to execute other non-verifiable programs. The *McManis* system includes a program executer that executes verifiable architecture neutral programs and a class loader that prohibits the loading and execution of non-verifiable programs unless the non-verifiable program resides in a trusted repository of such programs or the non-verifiable program is indirectly verifiable by way of a digital signature on the non-verifiable program that proves the program was produced by a trusted source.

Claim 1 recites receiving automatically authenticated code, the automatically authenticated code including an embedded first hash value of the unauthenticated code. A first hash value of unauthenticated code, such as for example native code, is embedded into automatically authenticated code, such as for example JAVA code that references the native code. The automatically authenticated code is received and the unauthenticated code is received; the automatically authenticated code and unauthenticated code are two separate entities. The method to verify the unauthenticated code comprises generating a second hash value from the received unauthenticated code and comparing the generated second hash value of the received unauthenticated code with the embedded first hash value from the automatically authenticated code. McManis does not teach or suggest the features as recited in claims 1, 11, and 21.

In the rejection of claims 1, 11, and 21, the Office Action refers to the following portion of *McManis*:

> However, in this case, the Compiler's and CompParty's public keys are retrieved from the trusted key repository 106 and respectively used to decrypt the MD_C and HashFunction$_C$ ID in the DigitalSignature$_C$ and the MD_{CP} and the HashFunction$_{CP}$ ID in the DigitalSignature$_{CP}$. Furthermore, the test message digests (TestMD_C and TestMD_{CP}) corresponding to the decrypted MD_{CP} and MD_C are generated by computing hash codes on the data bits of the ASProgram code plus the DigitalSignature$_{CP}$ for the TestMD_C and on the same data bits plus the DigitalSignature$_C$ for the TestMD_{CP}, according respectively to the HashFunction$_C$

and $HashFunction_{CP}$ identified by the decrypted $HashFunction_C$ ID and $HashFunction_{CP}$ ID.

If the $DigitalSignature_C$ and/or the $DigitalSignature_{CP}$ is not verified (i.e., MD_C .noteq.$TestMD_C$ and/or MD_{CP} .noteq.$TestMD_{CP}$) for every ASProgram, then the signature verifier 136 sends back to the class loader 136 a failed result. In response, the class loader aborts the class loading procedure and generates an appropriate message that this has occurred.

However, if the $DigitalSignature_C$ and the $DigitalSignature_{CP}$ are both verified (i.e., MD_C =$TestMD_C$ and MD_{CP} =$TestMD_{CP}$) for every ASProgram, then the ANProgram executer 124 again calls signature verifier 132 to verify the OrigParty's signatures ($DigitalSignature_{OP}$) for the ANPrograms from which the ASPrograms were compiled. To verify the OrigParty digital signatures, the $DigitalSignature_{OP}$ of each is verified in the same manner as was discussed earlier in the section concerning compilation of the ANProgram 200.

McManis, column 10, line 64 through column 11, line 25.

This portion of *McManis* teaches that compiler and compiler party public keys are retrieved from a trusted repository and used to decrypt compiler and compiler party digital signatures located in unauthenticated code, i.e. the application specific program (ASProgram) to obtain the values for the message digests (MDs) and hash functions for the compiler (C) and the compiling party (CP). The hash function for the compiler that is identified are then used to hash the bits of the application specific program (ASProgram) plus the digital signature of the originating party to obtain a test message digest for the compiler. The hash function for the compiling party is then used to hash bits of the ASProgram plus the digital signature of the compiler to obtain a test message digest for the compiling party. A determination is then made as to whether either the message digest for the compiler does not equal the test message digest for the compiler or the message digest for the compiling party does not equal the test message digest for the compiling party. If either of these conditions are met, the message is not authenticated.

Thus, the sections of *McManis* that the Office Action alleges teaches the features of the present invention are actually directed to authenticating the digital signatures of the compiler and compiling party for application specific programs in an object class (see column 10, lines 32-42). This verification is based on information decrypted from unauthenticated code. In contrast, the claims of the present invention recite that a first hash value of unauthenticated code is embedded in automatically authenticated code.

McManis does not teach or suggest embedding a hash value of unauthenticated code in automatically authenticated code.

Additionally, *McManis* teaches comparing text message digests for the compiler and compiler party, generated using data bits of unauthenticated code (ASProgram), with the decrypted compiler and compiler party message digests from the same unauthenticated code (ASProgram). Thus, *McManis* teaches to compare a message digest encrypted in unauthenticated code with a message digest generated from the unauthenticated code. In contrast, claims 1, 11, and 21 recite a first hash value embedded in automatically authenticated code and comparing the first hash value with a second hash value generated from a separately received unauthenticated code. In other words, in the presently claimed invention, the first hash value is obtained from automatically authenticated code and the second hash value is obtained from separate unauthenticated code. In *McManis*, the message digests are decrypted and generated from the same code, i.e. the ASProgram. There is no automatically authenticated code in *McManis*.

In view of the above, Applicants respectfully submit that *McManis* does not teach each and every feature of independent claims 1, 11, and 21 as required under 35 U.S.C § 102(b). At least by virtue of their dependency on claims 1, 11, and 21, respectively, *McManis* does not teach each and every feature of dependent claims 2, 4-6, 12, 14-16, 22, and 24-26. Accordingly, Applicants respectfully request withdrawal of the rejection of claims 1-2, 4-6, 11-12, 14-16, 21-22, and 24-26 under 35 U.S.C § 102(b).

III. 35 U.S.C. § 103, Alleged Obviousness Based on McManis

The Office Action rejects claims 3, 10, 13, 20, and 23 under 35 U.S.C. § 103(a) as allegedly being unpatentable over McManis (U.S. Patent Number 5,692,047). This rejection is respectfully traversed.

As discussed previously, *McManis* does not teach or suggest the features as recited in claims 1, 11, and 21. Specifically, *McManis* does not teach automatically authenticated code, let alone automatically authenticated code that is a platform independent application or applet, as recited in dependent claims 3, 13 and 23. To the contrary, as discussed above, *McManis* performs operations based on only the data that is

encrypted in or derived from the unauthenticated code, i.e. the ASProgram. Therefore, at least by virtue of their dependency on claims 1, 11, and 21, respectively, *McManis* does not teach or suggest the features of dependent claims 3, 10, 13, 20, and 23. Accordingly, Applicants respectfully request withdrawal of the rejection of claims 3, 10, 13, 20, and 23 under 35 U.S.C. § 103(a).

IV. 35 U.S.C. § 103, Alleged Obviousness Based on McManis and Cordery

The Office Action rejects claims 7-9, 17-19, and 27-29 under 35 U.S.C. § 103(a) as allegedly being unpatentable over McManis (U.S. Patent Number 5,692,047) in view of Cordery et al. (U.S. Patent Number 6,480,831). This rejection is respectfully traversed.

As discussed above, *McManis* does not teach receiving automatically authenticated code, the automatically authenticated code including an embedded first hash value of the unauthenticated code. In addition, *McManis* does not teach comparing a first hash value that is embedded in automatically authenticated code with a second hash value that is generated from separately received unauthenticated code. Furthermore, Applicants respectfully submit that *Cordery* does not teach or suggest these features.

Cordery is directed toward a method and apparatus for securely transmitting keys from a postage metering apparatus to a remote data center. *Cordery* teaches a method for transmitting a key from a first device to a remotely located second device includes steps of generating the key within the first device; selecting one of a plurality of one-time pad values from a one-time pad stored within the first device; creating a hash of at least the key and the selected one of the plurality of one-time pad values; and sending the hash and the key from the first device to the second device. Cordery does not teach anything regarding automatically authenticated code or embedding hash values in automatically authenticated code. Moreover, Cordery does not teach or suggest anything regarding comparing a first hash value that is embedded in automatically authenticated code with a second hash value generated from separately received unauthenticated code. Since neither *McManis* nor *Cordery* teach or suggest these features, any alleged combination of *McManis* and *Cordery* still does not teach or suggest these features.

Thus, neither *McManis* nor *Cordery*, either alone or in combination, teach or suggest receiving automatically authenticated code, the automatically authenticated code including an embedded first hash value of the unauthenticated code as recited in independent claims 1, 11, and 21. At least by virtue of their dependency on claim 1, 11, and 21, respectively, neither *McManis* nor *Cordery*, either alone or in combination, teach or suggest the features of dependent claims 7-9, 17-19, and 27-29. Accordingly, Applicants respectfully request withdrawal of the rejection of claims 7-9, 17-19, and 27-29 under 35 U.S.C. § 103(a).

V. Conclusion

It is respectfully urged that the subject application is patentable over the cited references and is now in condition for allowance. The Examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the Examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

Respectfully submitted,

DATE: January 27, 2004

Stephen J. Walder, Jr.
Reg. No. 41,534
Carstens, Yee & Cahoon, LLP
P.O. Box 802334
Dallas, TX 75380
(972) 367-2001
Attorney for Applicants

SJW/vja